Java Training 101
Section 1

By
Suraj

Outline

- Fundamentals
- Hands-on work
- Design review
- Project selection

Fundamentals

- Java is a software-only platform
  - It runs on other hardware-based platforms
- There are 2 components to Java
  - Java Virtual Machine (JVM) - a software that can be ported onto different hardware platforms
  - Java Application Programming Interface (Java API) - code written in Java that runs on the JVM
- Packages
  - Helps organize classes
  - Helps resolve naming conflicts
- Classes
  - Programming equivalent of a real-world object, ex. Car
- Interfaces
  - A class that only contains method signatures, ex. start, stop, increaseSpeed
  - Defined by - "public interface Automobile"
- Methods
  - Main - optional
  - Constructors - optional
- Keyword: super - to invoke the superclass' constructor

Fundamentals (Cont'd)

- Pre-requisites (see http://bit.ly/java101-prereq)
  - Java is installed on your computer
  - The Eclipse IDE has been installed

- [Hands-on task] Write a simple command-line java program and execute it using the command line
  - Make sure you have java available to you. Type: *which java*
  - Use a text editor to type the code using Java language syntax
  - Compile the code into byte-code using the java compiler:
    javac <filename>.java
  - Execute the compiled program:
    java <classname>

```
package com.test;

public class HelloWorld {
    public static void main(String[] args) {
       System.out.println("Hello World ");
       //modify this to add date in round 2
    }
}
```

Fundamentals (Cont'd)

- [Hands-on Task] Write a simple command-line java program and execute it using the Eclipse IDE
  - Open the Eclipse IDE (that you installed following the steps in the pre-reqs)
  - Create a new project - call it "Test"
  - In that project, create a new Package - call it "com.test"
  - In that package, create a new Class - call it HelloWorld
  - Repeat creating the method from the previous (command-line) technique
- CLASSPATH -
  - Command line usage demo
    - Tomcat
    - Program that calls needs another jar file to execute
- Inheritance - inclusion of behavior of method in the base class in a derived class (for code re-use)
  - Implementation
  - Class Document - 2 methods (add and update)
  - Class Production Drawing
  - - extends Document
  - - has an update method that invokes parent update method + does other things

Fundamentals (Cont'd)

- Exception handing
  - Why?
    - Because problems can happen anytime
    - Best-practice technique
  - How?
    - Use the try -> catch -> finally construct
    - Use throws clause where needed. Think through the use cases and how an error needs to be handled first.
  - Code Demo
- Design Patterns
  - Described as a effectively proven solution to a problem that has occurred repeatedly (*Ref:* http://www.javacamp.org/designPattern)
  - Example: Singleton - one instance of a class used everywhere in the application
- Concurrency and synchronization
  - Process - a program or an application
  - Thread - a light-weight process. Each process has at least one.
  - ATM usage example discussion on how threading can impact transactions

Project Design

- SDLC
  - Requirements gathering
  - Analysis & Design
  - Development
  - Testing
  - Deployment
  - Support
- HOWTO Design
  - RUP - *Ref:* http://www-01.ibm.com/software/awdtools/rup/
    - Outline Actors and Use cases - use case diagram
    - Describe Use cases in detail
      - Name, brief description, detailed description, pre-requisites, step-by-step events, exceptions, post-condition
    - Describe objects, attributes, relationships - class diagram
  - [Hands-on activity] Create a design for a project selected